KAIST

CS 380

Introduction to Computer Graphics

LAB (5)

2015.04.01

VISUAL COMPUTING Lab

## Task 1: Rigid Body Transformation

KAIST

- Define RigTForm class
  - Translation t (3D point vector) and rotation r (4D Quaternion vector)
  - Rigid body transformation $A = TR$
- Implement manipulations
  - Inversion
  - Multiplication
  - Conversion to matrix
  - Conversion from a translation vector
  - Conversion form a quaternion
  - Multiplication with a vector
- Alternate Matrix4 class with RigTForm class in asst2.cpp.



VISUAL COMPUTING Lab

## Task 1: Rigid Body Transformation

KAIST

- RigTForm inversion

$$A^{-1} = (TR)^{-1}$$
$$= \left( \begin{bmatrix} i & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} r & 0 \\ 0 & 1 \end{bmatrix} \right)^{-1} = \begin{bmatrix} r & 0 \\ 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} i & t \\ 0 & 1 \end{bmatrix}^{-1}$$
$$= \begin{bmatrix} r^{-1} & -r^{-1}t \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} i & -r^{-1}t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} r^{-1} & 0 \\ 0 & 1 \end{bmatrix}$$
$$= T_{inv} R_{inv}$$

t = 3x1 vector
r = 3x3 matrix

- Vector multiplication $Ax = TRx$
  - In quat.h, quat-vector multiplication is already implemented.
  - For rotation multiplication, simply multiply r quaternion to x.
  - For translation, just add t to rotated x.



VISUAL COMPUTING Lab

## Task 1: Rigid Body Transformation

KAIST

- RigTForm multiplication

$$A_3 = A_1 A_2 = T_1 R_1 T_2 R_2$$
$$= \begin{bmatrix} i & t_1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} r_1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} i & t_2 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} r_2 & 0 \\ 0 & 1 \end{bmatrix}$$
$$= \begin{bmatrix} i & t_1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} r_1 & r_1 t_2 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} r_2 & 0 \\ 0 & 1 \end{bmatrix}$$
$$= \begin{bmatrix} i & t_1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} i & r_1 t_2 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} r_1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} r_2 & 0 \\ 0 & 1 \end{bmatrix}$$
$$= \begin{bmatrix} i & t_1 + r_1 t_2 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} r_1 r_2 & 0 \\ 0 & 1 \end{bmatrix}$$
$$= T_3 R_3$$



VISUAL COMPUTING Lab

## Task 2: Arcball

KAIST

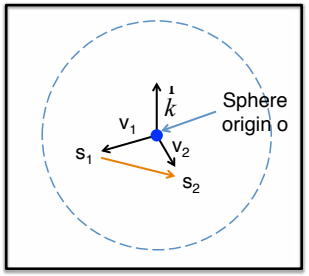
- Implement the arcball interface
  - Draw a sphere to represent the arcball
  - Implement an arcball function in openGL functions
- Compute rotation
  - Compute two 3D vectors on the screen space.
- Two helper functions
  - getScreenSpaceCoord: get the center of a sphere on screen cooridnate
  - getScreenToEyeScale: scale a sphere
- The radius of the sphere should be 0.25 * min (g_windowWidth, g_windowHeight).



VISUAL COMPUTING Lab

## Arcball Rotation

KAIST




- Sphere origion o – center of sphere, projection of a frame origin
- $s_1$ – clicked screen coordinate
- $s_2$ – dragged mouse screen coordinate

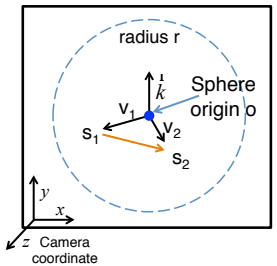


VISUAL COMPUTING Lab

## Arcball Rotation (hint)

KAIST

- $v_1$, $v_2$ – the directional vectors
- $v_{1x} = s_x - o_x$, $v_{1y} = s_y - o_y$, $v_{1x}^2 + v_{1y}^2 + v_{1z}^2 = r^2$






VISUAL COMPUTING Lab

## Task 3: Translation Fix Up

KAIST

- Translate the object as same as the mouse movement.
- Use g_arcballScale.
- Wherever the object is, the object should follow a mouse pointer.



VISUAL COMPUTING Lab